FINAL REPORT

Better Password Generator

Presentation Slides:
https://docs.google.com/presentation/d/1aMeJod_a9c_ZjRfooe8VbYQVl8F53HANDNAMG-rIyZw/edit?usp=sharing

Code for GPT-2 implementation:
https://drive.google.com/file/d/1EoKybE3ro2PyaPuGNJaXhyzY2ND7dlcA/view?usp=sharing

Code for Testing our scheme:
https://drive.google.com/file/d/1XqLoD-5SlWlvwBhtEv1M9hoajSHMQQOF/view?usp=sharing

Instructions on how to run the files:
https://drive.google.com/file/d/136JYPGAwnrOJbe9ow7iYVoFexakoE3dv/view?usp=sharing

Introduction

Better passwords are essential to the security of computer and information systems. Passwords control a large amount of user information, and a password leak can cause adversaries to get access to highly sensitive data. Password managers act as a great resource to generate and store better, less predictable passwords. However, they are a single point of failure and often rely on trusting a third party to reliably store and provide access to encrypted data. Additionally, a large majority of users still rely on a small number of weak, predictable passwords that they might have come up with. In our project, we aim to identify the current mechanisms and user patterns in password generation, analyze their robustness to attacks, and suggest a better mechanism to generate passwords that is easy to manage as well.

In this report, we will detail and implement a novel password generator based upon GPT-II. We also show that there is likely sufficient entropy in these passwords and that they are more memorable than existing solutions.

Background

There has been extensive research on password generation, management, and security. In a paper by Zviran and Haga, they surveyed users' passwords based on four factors: length (number of characters), composition (letters vs numeric), lifetime (how often the password is changed), and selection (how the password is generated - personal detail vs memorable string). They conducted a t-test and determined that of these, length does not affect memorability, while the other 3 factors do.

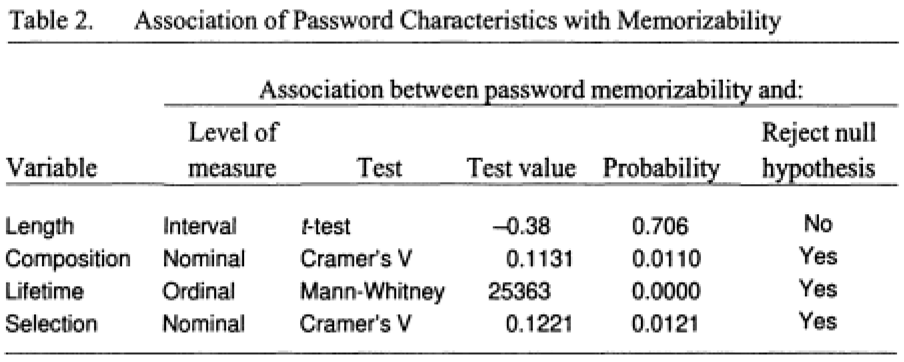


Table 2. Association of Password Characteristics with Memorizability

	Association between password memorizability and:				
Variable	Level of measure	Test	Test value	Probability	Reject null hypothesis
Length	Interval	*t*-test	–0.38	0.706	No
Composition	Nominal	Cramer's V	0.1131	0.0110	Yes
Lifetime	Ordinal	Mann-Whitney	25363	0.0000	Yes
Selection	Nominal	Cramer's V	0.1221	0.0121	Yes

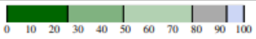
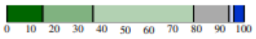
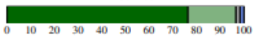
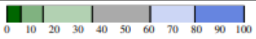
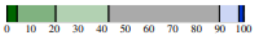
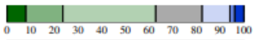
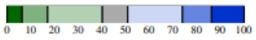
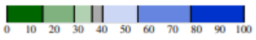
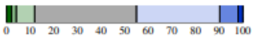
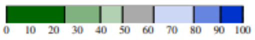
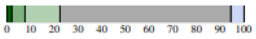
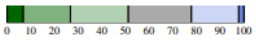
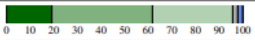
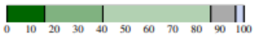
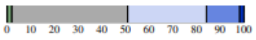
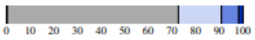
Despite these findings that length does not contribute to a decrease in memorability, the paper found that the average password is only 6 characters long, despite that being one of the most important factors for password security.

We also examined the characteristics that are often exploited by attackers. The findings can be summarized by Gehringer, who said that a good password does not:

- Contain dictionary words
- Be the name of a friend, relative, or a famous person
- Be less than 8 characters in length
- End with, start with, or contain a predictable sequence of letters like 123

Additionally, as mentioned in Gehringer's paper, the practice of implementing 'Lifetime Lockouts' should be avoided as it requires users to change passwords frequently which forces users to use less memorable passwords that they have no choice but to write down. Writing down passwords is a threat to the security of a password and hence must be avoided. Alternatively, users might also start coming up with very simple, less secure passwords which are again less secure.

Despite there being much literature describing good passwords and what to avoid, it turns out that the average person is not very good at determining what is and is not a good password. In a paper by Ur et. al., they compared perceived and actual password security, asking people to choose which of two passwords they thought would be more secure. Participants correctly recognized adding symbols and adding obscure words adds security.

PW_1	PW_2	Actually Stronger	Perceived Stronger	p	Perceptions
p@ssw0rd	pAsswOrd	PW_2 (4×10^3)	PW_1	<.001	
punk4life	punkforlife	PW_2 (1×10^3)	PW_1	<.001	
1qaz2wsx3edc	thefirstkiss	PW_2 (3×10^2)	PW_1	<.001	
iloveyou88	ieatkale88	PW_2 (4×10^9)	Neither	–	
astley123	astleyabc	PW_2 (9×10^5)	Neither	–	
jonny1421	jonnyrtxe	PW_2 (7×10^5)	Neither	–	
brooklyn16	brooklynqy	PW_2 (3×10^5)	Neither	–	
abc123def789	293070844005	PW_2 (8×10^2)	Neither	–	
puppydog3	puppydogv	PW_2 (7×10^2)	Neither	–	
qwertyuiop	bradybunch	PW_2 (4×10^2)	Neither	–	
bluewater	nightgown	PW_2 (3×10^1)	Neither	–	
iloveliverpool	questionnaires	PW_2 (2×10^1)	Neither	–	
L0vemetal	Lovemetal	Neither	PW_1	<.001	
sk8erboy	skaterboy	Neither	PW_1	<.001	
badboys234	badboys833	Neither	PW_2	.001	
jackie1234	soccer1234	Neither	PW_2	.034	

🟩 PW_1 much more secure 🟩 PW_1 more secure 🟩 PW_1 slightly more secure ⬜ Equally secure 🟦 PW_2 slightly more secure 🟦 PW_2 more secure 🟦 PW_2 much more secure

Table 5. Pairs of passwords for which participants' perceptions of the relative security of the passwords differed from actual security. The number in parentheses indicates how many times stronger PW_2 was than PW_1 (ratio of guess numbers).

On the other hand, they underestimated the ability of password-cracking tools, which know about frequent substitutions (*punk4life vs punkforlife*), as well as the tendency of users to add numbers at the end of passwords (*astley123 vs astleyabc*). They also failed to recognize the power of unexpected capitalization (*pAsswOrd vs p@ssw0rd*) as well as the insecurity of keyboard patterns (*1qaz2wdc3edc, qwertyuiop*). Additionally, participants also did not realize that the popularity of words and phrases has an effect (*ieatkale188 vs iloveyou88*). As one astute participant noted, "eating kale is a lot more rare than love".

This paper shows that the heuristics that people use to determine password security often do not match reality, leading to users generating poor passwords on their own, and displaying the need for a good, easy-to-use password generator.

Threat Model

We considered a two-pronged threat model for this project. On one hand, we considered existing adversarial attacks like brute force attacks, dictionary attacks, and credential stuffing attacks. These attacks informed us about how much randomness our password generation should have, as well as reiterated the importance of having a large variety of random passwords.

On the other hand, we considered lazy users another threat to password security overall. We define lazy users as internet users with at least some of the following characteristics: they reuse passwords, generate their own passwords, conform only to required password requirements, require low-effort password tooling, and/or do not have a strong understanding of security concepts. Given the practical nature of our project, we will primarily focus on reducing the risk of lazy users on overall password security.

Brute Force Attacks

In a brute force attack, an adversary tries all possible passwords until the correct one is found. Brute force attacks take a long time and require a password verifier to check if the password is correct. Common defenses against these attacks include using long passwords and limiting incorrect password attempts.

Dictionary Attacks

A dictionary attack is similar to a brute force attack but uses a dictionary of common words in passwords instead of trying all possible combinations. Dictionary attacks are faster than brute force attacks but can be defended against by using obscure passwords unlikely to be added to the adversary's dictionary (ex. a randomly-generated string of characters).

Phishing

Phishing is a form of social engineering that directs users to an adversarial-controlled platform (website, phone call, email address) that impersonates another platform and attempts to get users to enter their credentials. Defenses against phishing attacks include spam filters, browser blacklists, and phishing education and awareness campaigns.

Credential Stuffing

Credential stuffing attacks exploit users who reuse passwords across multiple accounts by attempting to authenticate into one platform using credentials stolen from another platform. Credential stuffing attacks frequently try different variations on usernames and passwords to account for slight modifications. Defenses against credential stuffing include regularly changing passwords, using different passwords for different accounts, as well as limited login attempts.

Lazy Users

Lazy users often expose themselves to the above adversarial attacks by following poor password hygiene practices like reusing passwords, using low-entropy passwords, or falling for phishing attacks. Despite the existence of strong password security tools like password managers and password generators, they are not widely used because they are not easy enough to use. For example, password managers may not integrate well into browsers or mobile devices, and cannot be used easily on devices other than the user's own devices or when the user does not have access to the manager, disabling the user from authenticating in such circumstances. Password generators typically require using another application or website. While security-conscious users will happily accept the increased effort for heightened security, lazy users will not. To defend against lazy users, password tooling like managers and generators needs to be made easy to use. Browser and mobile integration of password managers and password generators have aided, but not solved, this problem.

User Survey Results

In addition to generating better passwords, it is also extremely important that users feel motivated to use these passwords generated ‘online’. Various password generators exist today. However, very few people seem to be using them.

Given the application-based focus of the project, we decided to conduct a user survey to learn more about user password practices, including password generation, password reuse, and password manager use. To improve on these practices, we also asked users why they did not use better password hygiene. Overall, we found that users follow mediocre password practices, and these practices are held back the most by the inability to remember more secure passwords and a higher amount of effort required to generate and store a diverse set of random passwords. Below are some highlights from the survey:

- 70% of users generate their own easy-to-remember passwords
- Most users only change their passwords when required or infrequently
- Most users have some variety of passwords, but not a unique password for all accounts
- Commonly cited reasons for not using a password generator or a large variety of passwords are a higher level of effort and inability to remember these passwords
- Most users do not use a large variety of passwords for memorability

Implementation:

Our better password Generator

Based on the results of our ‘User Survey’ and the current literature on this topic, We concluded that the best generated passwords would be ones that use a mnemonic or a decently coherent but memorable sentence. This sentence can later be manipulated to generate the actual password and make it more robust to attacks such as Brute Force Attacks and Dictionary Attacks. We used GPT-2 to generate these sentences.

GPT-2, a language model, can generate sentences of various forms. These sentences can be meaningful or meaningless, depending on the parameters fed into the model, specifically, ‘temperature’ and ‘top_k’. The ‘temperature’ parameter relates to the Boltzmann distribution - higher temperature means more random, as the temperature approaches 0, deterministic and repetitive. The ‘top_k’ parameter is the number of words considered for every step, and - 1 would mean deterministic. We attempted to set these parameters such that the generated sentences were not meaningful, but still coherent in a memorable way

Some Examples of generated sentences are as follows:

- 'todos', 'rarities', 'which', 'find', 'searching', 'that'
- 'wasp', 'story', 'bizarre', 'serenity', 'that', 'this'

Creating the password from the Generated Sentence:

- Generated Sentence: “Hello this days shall funny neither ...”, Random Positions: (1, 3)
- Result password: “elhsashlunet”
- *STEPS*:
    - Get the first 6 words of the sentence
    - Get the letters at the specified random positions in each word(0 indexed)
    - String together the letters obtained to get the password
    - *RESULT*: A 12 letter password that is decently unpredictable and memorable

We decided to only include small letters(a-z) in our sentences and hence our passwords as a default. There are ways that we used to add complexity to the passwords, but they were added as parameters that the user can choose to use or not to use to keep the password memorable for all users. For the random positions, the first position is restricted to 1-3, and if the last position overflows, the user is instructed to use the last letter of the word.

As discussed above, we made two refinements to our basic or default generator. First, we added the option to capitalize one word in the sentence which will show up as 2 capital letters in the generated password, which will increase the entropy of the password. Second, we added the option to include digits in the password. If the user indicates that they wish to do so, we tell the user to include a random digit before a certain number of words in the sentence.

For example, say the generated sentence is “Hello this days shall funny neither ...”.

The program will say something like this:
“Add digit 4 before words [1, 2, 6]”

Hence, the resulting password will be: “4el4hsashlun4et”

Per the project description, we wish to achieve no less than 50 bits of entropy. An ideal password generator with 50-bits of entropy would output one of 2^{50} passwords randomly, with each password having an equal probability. Since our password generator is hindered by the letter distribution of the English language, we will target well above 2^{50} possible passwords to achieve the same level of entropy as a randomly generated 50-bit string. Below, we’ll calculate the total number of possible passwords from our proposed design, including some of our proposed variations. Each variation of our passwords provides more than 2^{50} possible passwords, with our most secure option providing 2^{93} possible passwords. With this, even with the assumption that only the 10 most frequent characters in the English language are generated, we achieve 73 bits of entropy, well above our target.

6 words, 12 letters, lowercase: $26^{12} \approx 2^{56}$

6 words, 12 letters, randomly cased: $52^{12} \approx 2^{68}$

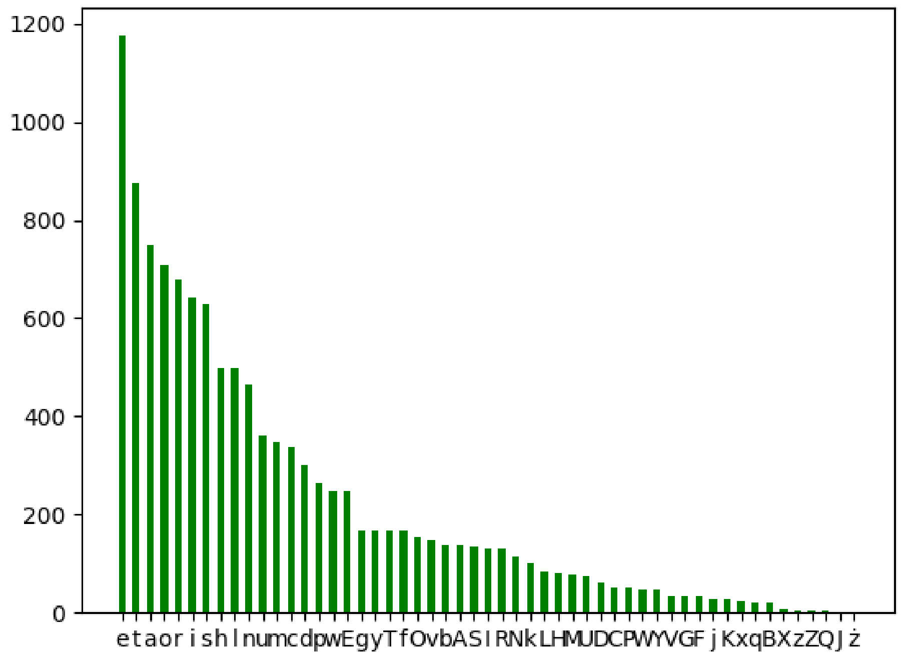
6 words, 12 letters, lowercase, 1-4 numbers inserted: $\sum_{i=1}^{4} 7^{i}10^{i}(26^{12}) \approx 2^{81}$

6 words, 12 letters, randomly cased, 1-4 numbers inserted: $\sum_{i=1}^{4} 7^{i}10^{i}(52^{12}) \approx 2^{93}$

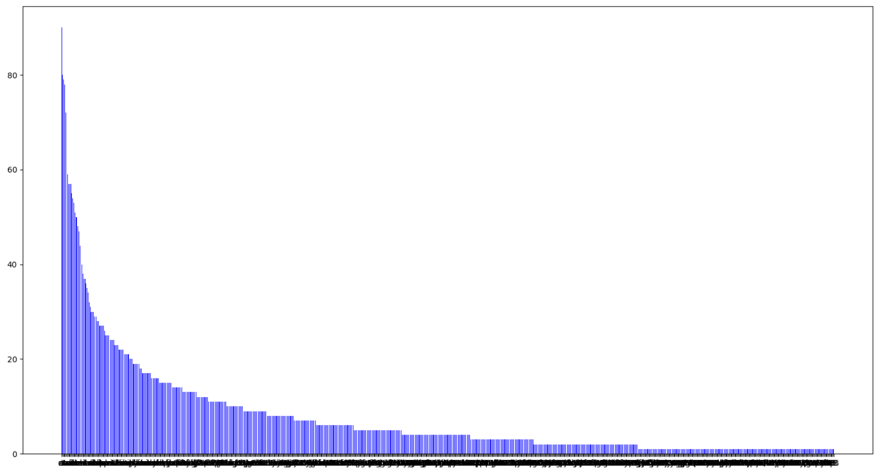
6 words, 12 letters, randomly cased, 1-4 numbers inserted using only the 10 most frequent characters: $\sum_{i=1}^{4} 7^{i}10^{i}(20^{12}) \approx 2^{73}$

To test the properties of our password generator, we conducted Monte Carlo experiments where we generated 1000 passwords with the same starting seed, “The”, temperature of 2.0, and top_k of 50.

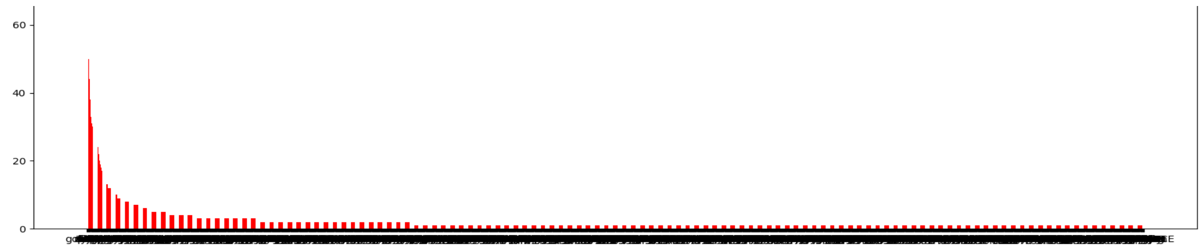
The first metric we examined was the frequencies of letters, which roughly matched the frequencies of letters in text. This is to be expected since GPT-II attempts to emulate human writing.



Next, we examined the frequencies of bigrams that come from a single word (e.g. ‘el’ from “Hello”). The most frequent bigram appeared 90 times.



We also recorded the frequencies of words that appear. The most frequent word appeared 50 times out of 6000 total words.



This last graph displays an extremely long tail signifying that many many words appeared only once, giving us hope that an attacker would not be easily able to replicate an individual password generation. Additionally, since all of these runs were done with the same initial seed, it is likely that with further improvements such as by randomizing the initial seed, we can further flatten this distribution.

Conclusion:

Passwords are an integral part of daily life today. However, most users do not follow best practices for secure password generation and create memorable passwords that are too short, contain dictionary words, use personal information, and follow common password patterns. Such user practices indicate that it is imperative to popularise the use of password generators to enable more security on the internet. Many password generators already exist, however are not as prevalent as they should be considering the benefits they provide. Our user study suggests that users choose not to use password generators because the passwords generated using these generators are difficult to remember and thus hard to use.

A better password generator should thus increase memorability while maintaining high levels of randomness and entropy. The scheme explored in our project does so by using the GPT-2 language model. In our implementation, we decided that GPT-2 needed some pseudo-random modifications to achieve better passwords. With the modifications, we achieve greater than the target 2^{50} bits of entropy.

The efficacy of this scheme is still unclear. Even though we achieve an entropy much larger than the required 2^{50}, the randomness and entropy of the scheme require more evaluating as the English language itself is not random and follows certain patterns which could be exploited. Moreover, a user-survey testing the memorability of passwords under this scheme would help to evaluate the memorability of the new scheme; the scheme depends on the assumption that providing a GPT-2 derived sentence as a mnemonic can improve memorability. Future research can build off of our implementation by taking a more scientific approach to parameters and input-text choices of the model can be had as well.

References:

- Study about characteristics of passwords
  - https://www.tandfonline.com/doi/pdf/10.1080/07421222.1999.11518226?casa_token=GOKGnaE18YoAAAAA:imk7qH1AUDPPlUYBZO8UmSAUw4TzdMgM3QojQJxe_F0BFz_yzCDPkx1gqF2hqxZrB0oKncsJ-yoAAA
- Do Users' Perceptions of Password Security Match Reality?
  - https://dl.acm.org/doi/pdf/10.1145/2858036.2858546
- How to ensure effective password security
  - https://dl.acm.org/doi/pdf/10.1145/508171.508195?casa_token=FU5ym6VCqfcAAAAA:pzWGtxD9CUp-ZE4GJkvJS4GT_B6uiQNUAV_dPzws_XggJalUOGZsYa1zggkRifppXy0tWQ8b-1RDcg
- Memorization techniques:
  - https://learningcenter.unc.edu/tips-and-tools/enhancing-your-memory/
- Choosing passwords: Security and Human Factors:
  - https://www.researchgate.net/publication/3955154_Choosing_passwords_Security_and_human_factors
- Encouraging users to improve passwords:
  - https://link.springer.com/article/10.1007/s10207-019-00429-y
- Attacks:
  - https://www.researchgate.net/publication/236898951_A_Survey_of_Password_Attacks_and_Comparative_Analysis_on_Methods_for_Secure_Authentication
- GPT-2:
  - https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf